MSU Latent AFIS[1]

User Guide v1.6

Kai Cao, Luan Nguyen, Cori Tymoszek, and Anil Jain
Michigan State University

November 21, 2019

1 Executive Summary

This document provides User Guide to MSU Automated Latent Fingerprint Identification System (AFIS). The end-to-end system was designed and prototyped with a R&D contract from IARPA (contract no. 2018-18012900001). The system utilizes deep networking models and dictionary-based methods. An overview of the architecture is shown in Figure 1. The system has been configured to run on a multi-core platform and has been evaluated on four latent databases: NIST SD27, MSP, WVU and N2N. Its performance on NIST SD27 with a 100K gallery is 65.7%. A multi-core solution implemented on Intel(R) Xeon(R) CPU E5-2680 v3@2.50GHz takes 1ms to compare one latent with one rolled print. Hence, a latent search against 100K reference prints can be completed in 100 seconds. Latent feature extraction time is 15 seconds on a machine with Intel(R) i7-7780@4.00GHz (CPU) and GTX 1080 Ti (GPU).

2 General Setup

This software was designed for Linux systems only. It has been tested with Ubuntu versions 16.04 and 18.04.

This system uses Python, as well as a number of Python packages that require specific versions for compatibility. An environment manager such as Conda is recommended to properly isolate the Latent AFIS system. This helps to ensure compatibility, and prevents interference with any other packages that may already be installed.

We recommend Miniconda for Python 2.7. It can be downloaded here:

`https://conda.io/miniconda.html`

Install Miniconda according to the developers instructions.
During installation, you will likely be asked about the default installation directory, and whether you would like to append conda to your `.bashrc` file (or something similar). It is fine to answer yes to both questions (unless you have some other preference).

[1]K. Cao, D. Nguyen, C. Tymoszek, A. K. Jain, "End-to-end Latent Fingerprint Search", MSU Technical Report MSU-CSE-18-3, 2018 (under review, IEEE Trans. Information Forensics and Security)
https://arxiv.org/pdf/1812.10213.pdf

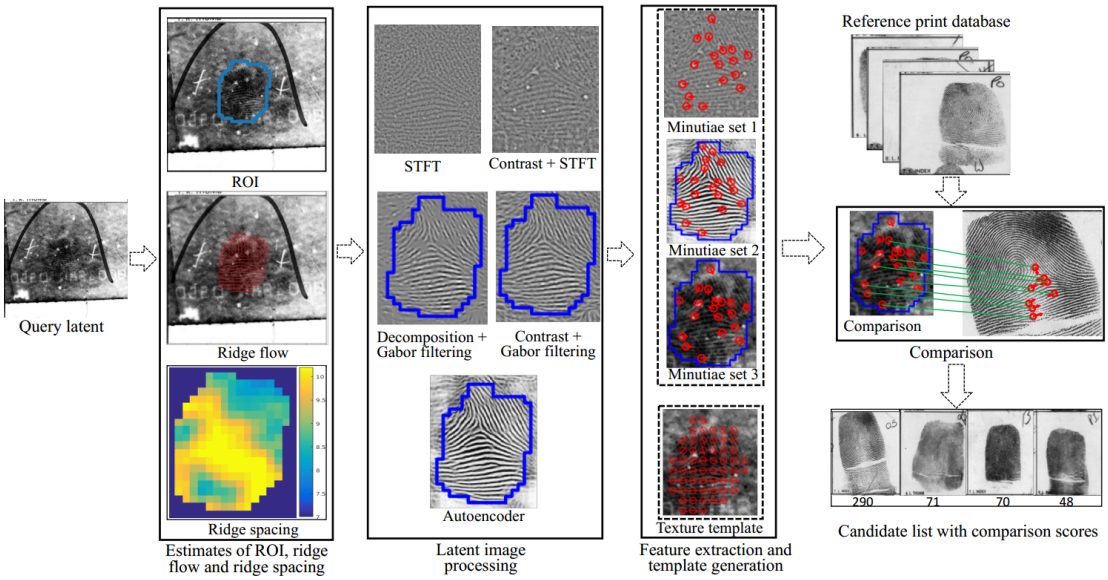


Figure 1: Overview of the Latent AFIS. Given a query latent, three minutiae templates and one texture template are generated. Two matchers, i.e., minutiae template matcher and texture (virtual minutiae) template matcher are used for comparison between the query latent and reference prints.

A NOTE ON CONDA:

Miniconda is a special variation of the environment and package manager Anaconda. The difference between Miniconda and Anaconda is that Anaconda comes with a large selection of Python packages, and Miniconda comes with none. Since we specify all needed packages, it is not necessary to install Anaconda and its default library of packages. This is faster and saves storage space. It should be noted that aside from the initial package installation, the usage for Miniconda and Anaconda is identical, and the names Miniconda, Anaconda, and conda are used interchangeably, both in this document and generally.

After Miniconda is installed, copy the `LatentAFIS` folder to local storage. In the root of this folder, there is a file called `environment.yml`, which specifies all the needed packages and their respective versions. Conda can set up a new environment according to these specifications with a single terminal command. Open a terminal window, and navigate to the `LatentAFIS` folder. Use this command to create the environment and install the packages:

```
conda env create -n latentafis -f environment.yml
```

The option `-n latentafis` specifies the name of the environment, in this case latentafis. You may use a different name if you prefer.

After conda is finished creating the environment and installing the packages, activate the new environment:

```
conda activate latentafis
```

On some systems, you may need to use:

```
source activate latentafis
```

If the environment is successfully activated, the terminal prompt will change from this:
`user@pcname: $`
To this:
`(latentafis) user@pcname: $`
For the remainder of this document, all terminal commands are given under the assumption that the conda environment is active. At this point, two final packages need to be installed: OpenCV and Tensorpack. OpenCV can be installed through conda as well, but it is recommended to install it through pip instead, as the conda version of OpenCV is quite slow. In the case of Tensorpack, the correct version is not available through conda. To install through pip, use these commands:

```
pip install opencv-python==3.4.2.17
pip install tensorpack==0.8.9
```

This concludes the environment creation.

2.1 Configuration

In the `LatentAFIS` folder, there is a file called `afis.config`. This file contains a number of settings, some of which are critical, and some of which are simply for convenience. Open this file in a text editor and change it as needed. The two most important changes for initial setup are:

1. Modify the paths to match the current install location of the software. For example, the paths by default will be in the general form `/home/LatentAFIS/folder1/`. Change the `home` part of the path to wherever the `LatentAFIS` folder is actually located on your system.

2. Change the Latent and Gallery directories as needed. They are set by default to point to a `data/` folder which contains any templates and images used by the software. You may follow this default structure, or use a different one if you choose - just be sure to update this configuration file. The default file structure is illustrated by the `sample_data/` folder.

3 Models

NOTE: Scripts to train new models are not included in the initial version of the software. These will be provided in the future. The software includes previously trained models that are sufficient for initial evaluation.
Three models (enhancement, minutiae detection, and descriptor extraction) are needed to process images for use as the probe (latent query) or in the gallery (database of reference prints). The architecture of these models is shown in Figure 2. Pre-trained models are provided which represent the latest stable version of the models. However, if modifications are necessary, these models can be trained again.

4 Template Extraction

The above models are used to transform an image file (e.g. JPEG, BMP, PNG) to a template. The generated template contains information about the minutiae points, their associated descriptors, and other features of the fingerprint contained in the photo. This can be done on the fly by the Matching module for the input image, but it is also necessary to complete this process separately to generate the gallery of reference templates. There are two scripts to do this extraction - one

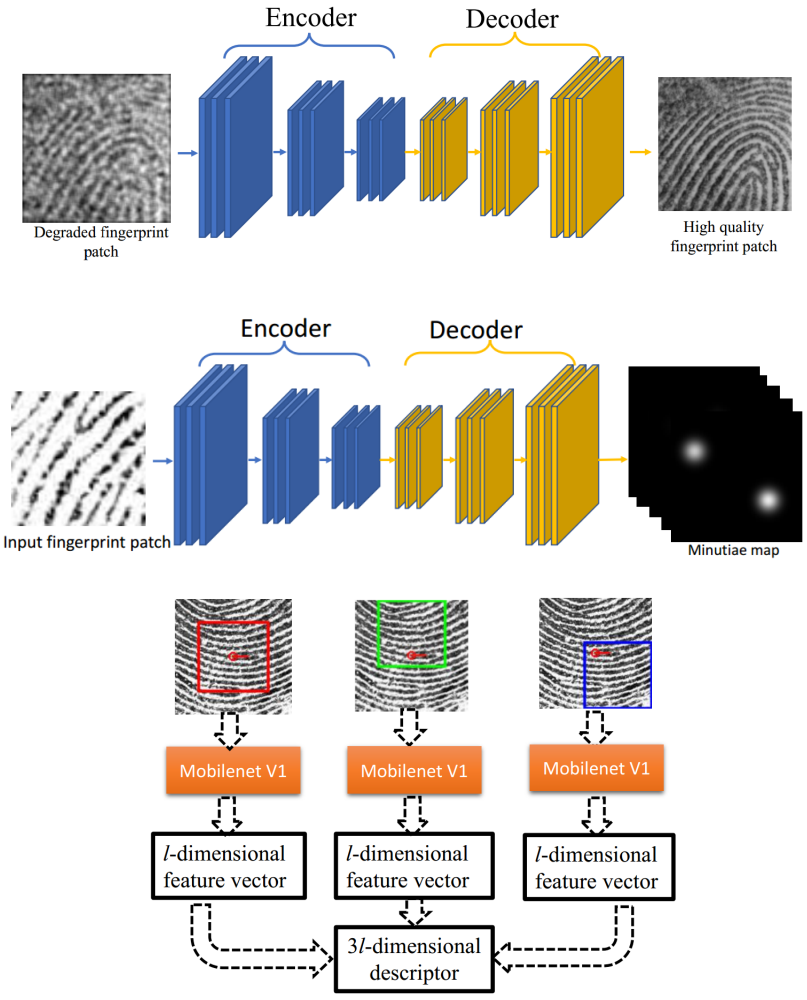


Figure 2: The architecture diagrams for the three models: Image enhancement (top), minutiae detection (middle), and descriptor extraction (bottom).

for the latent images (`latentafis/extraction/extraction_latent.py`), and one for the rolled reference images (`latentafis/extraction/extraction_rolled.py`). The general format of the usage is the same for both:
`python scriptname.py --flag1=/path/to/a/folder/ --flag2=/path/to/another/folder/`
Available flags for both `extraction_latent.py` and `extraction_rolled.py` are:

- `--idir=/path/to/image/folder/`: This flag points to a directory where all the images inside should be processed.
- `--tdir=/path/to/template/folder/`: This flag points to a directory where all the generated templates should be saved. Note that the software is configured to overwrite existing templates at this location in the case of name conflicts.

Additionally, the flag `--i==/data/images/filename.bmp` is available for `extraction_latent.py`

in the case of a single latent query that needs to be processed. This flag is used in place of `--idir`, and requires a path to a single image instead of a folder. It is required that either `--i` or `--idir` is used to specify the input image(s), but `--tdir` is optional and will be taken from the afis.config file if omitted.
A small subset of images and templates (3 latents and the 3 corresponding rolled mates) are located in the `sample_data/` folder. It is recommended that users verify that the software is working correctly by running the Feature Extraction scripts on the sample images. Please be sure to set the template directories where the generated templates will be stored - if they are set to the same location as the existing sample templates, the software will fail.
To verify that the templates generated match the samples, it is recommended to compare the size of the template files. For example, the latent image 001.bmp from the NIST SD27 dataset should always generate a template file of size 980.1 kB. If the sizes match, the templates should be evaluated further by performing matching/identification.

5 Matching

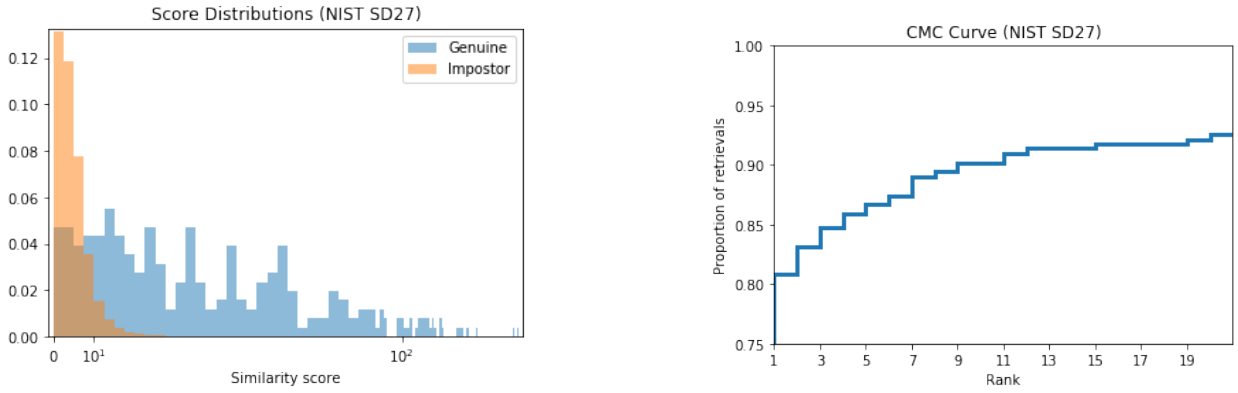


Figure 3: Score distribution (left) and CMC curve (right) generated by the system for NIST SD27, with background size 258.

The C++ code to perform matching/identification is located in `latentafis/matching/`. First, the code must be compiled with the `make` command. Second, we need to configure the location of the included libraries:
`sudo ldconfig /somepath/LatentAFIS/matching/boost/stage/lib`
where `/somepath/` is replaced by the location of the LatentAFIS folder. After that, the usage is as follows:
`./match -flag /path/to/something`
The flags available for the matcher are:

- `-l`: This flag is analogous to the `--i` flag in the feature extraction scripts. It specifies the path to a single .dat file for the latent query template.
- `-ldir`: This flag is analogous to the `--idir` flag in the feature extraction scripts. It specifies the path to a directory of latent .dat files for batch matching. The matcher will generate one score file for each latent in the directory.

- `-g`: This flag specifies the path to the directory containing the templates (as .dat files) for the rolled gallery.
- `-s`: This flag specifies the path to the directory where score files should be saved. Note that these will overwrite any existing score files at this location.
- `-c`: This flag specifies the path to a codebook file needed by the matcher. The codebook is provided, and should be located in the same folder as the matcher. The file name should be something like `codebook_EmbeddingSize_96_stride_16_subdim_6.dat`.

All of these flags are optional, and the corresponding information will be taken from the `afis.config` file if omitted. However, the `-l` flag does not have a config entry - if both this and the `-ldir` flags are omitted, the software will default to use `-ldir`, and will use the corresponding directory in `afis.config` for batch latent matching. Also, please note that Python flags (for feature extraction above) used the format `--f=data`, whereas C++ flags (for the matcher) use the format `-f data`.

By default, the results will be saved at `latentafis/scores/`, where the matcher creates one .csv file for each query image. In this file, there is one row for each candidate in the gallery of reference prints, and the similarity score is listed for each candidate.

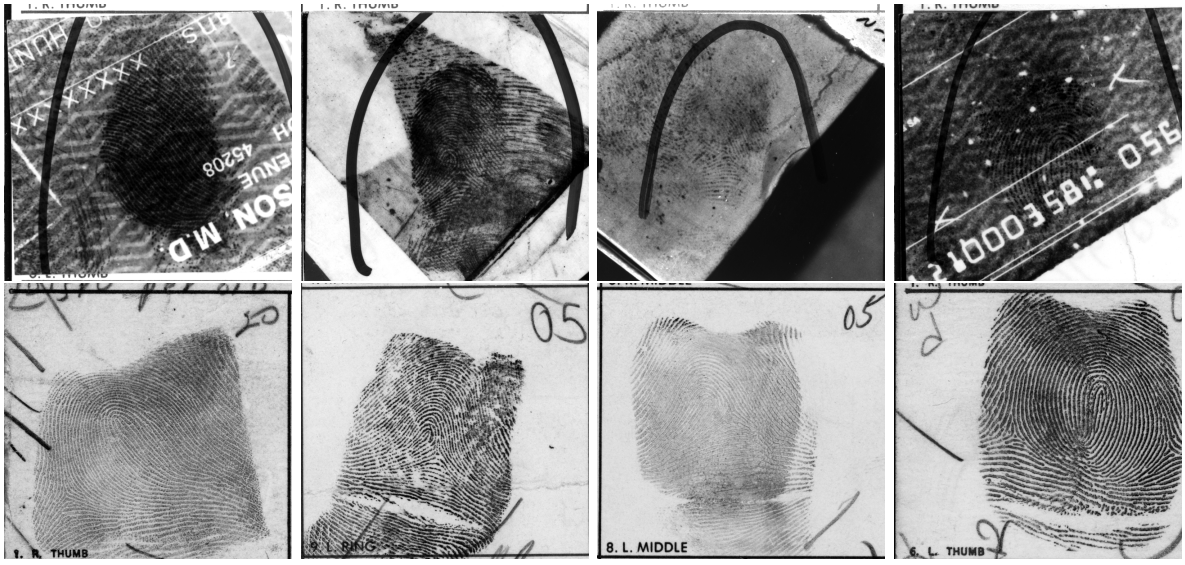
Please note that these scores are not percentages, and they do not have an explicitly defined upper bound. They are most informative when analyzed relative to one another, i.e. compare the magnitude of the rank 1 score with that of rank 2, 3, etc. For reference, when matching the NIST SD27 dataset where the background is of size 258 and consists of the true mates of each of the 258 latents, the average genuine score is 44.5, and the maximum genuine score is 254.19. The genuine and impostor distributions for NIST SD27, as well as the CMC curve, are shown in Figure 3. Additional reference pairs from both successful and failed queries are shown in Figure 4.

6 Graphical User Interface

For data-intensive tasks, such as evaluation of the software on a large dataset, we recommend using the command line operations detailed above. However, we have also provided a graphical user interface (GUI) to give additional insight into the feature extraction and matching process. Features of this GUI application:

- End-to-end feature extraction and search of a latent query image against a gallery
- Visualization of the features extracted from the latent, including image enhancements, foreground segmentation (or cropping), and minutiae templates
- Template generation and enrollment of a gallery of rolled reference prints
- Visualization of the search results in the form of a ranked candidate list, along with the similarity scores for each candidate mate
- Visualization of the minutiae correspondence between a query latent and a candidate mate

Usage of the GUI requires the same setup and prerequisite packages described in Section 2. Instructions for installing the GUI itself are located in the README file in the directory `LatentAFIS/latentafis-gui`. For a visual demonstration of the GUI's features, a video is included in the software's root directory.



(a) Success case: rank 1, similarity score 79.219.

(b) Success case: rank 1, similarity score 118.411.

(c) Failure case: rank 25, similarity score 8.483.

(d) Failure case: rank 5, similarity score 7.252.

Figure 4: Example pairs from NIST SD27 with identification results from the Latent AFIS against a background of size 258. Each latent print in the top row is the true mate of the corresponding rolled print in the bottom row. The rolled mates in pairs (4a) and (4b) were successfully returned at rank 1. Those from pairs (4c) and (4d) were not.

Warranty

This software is provided as a companion to our publications, and as a reference point for other groups wishing to do similar research. This software is provided as-is with no warranty or maintenance.